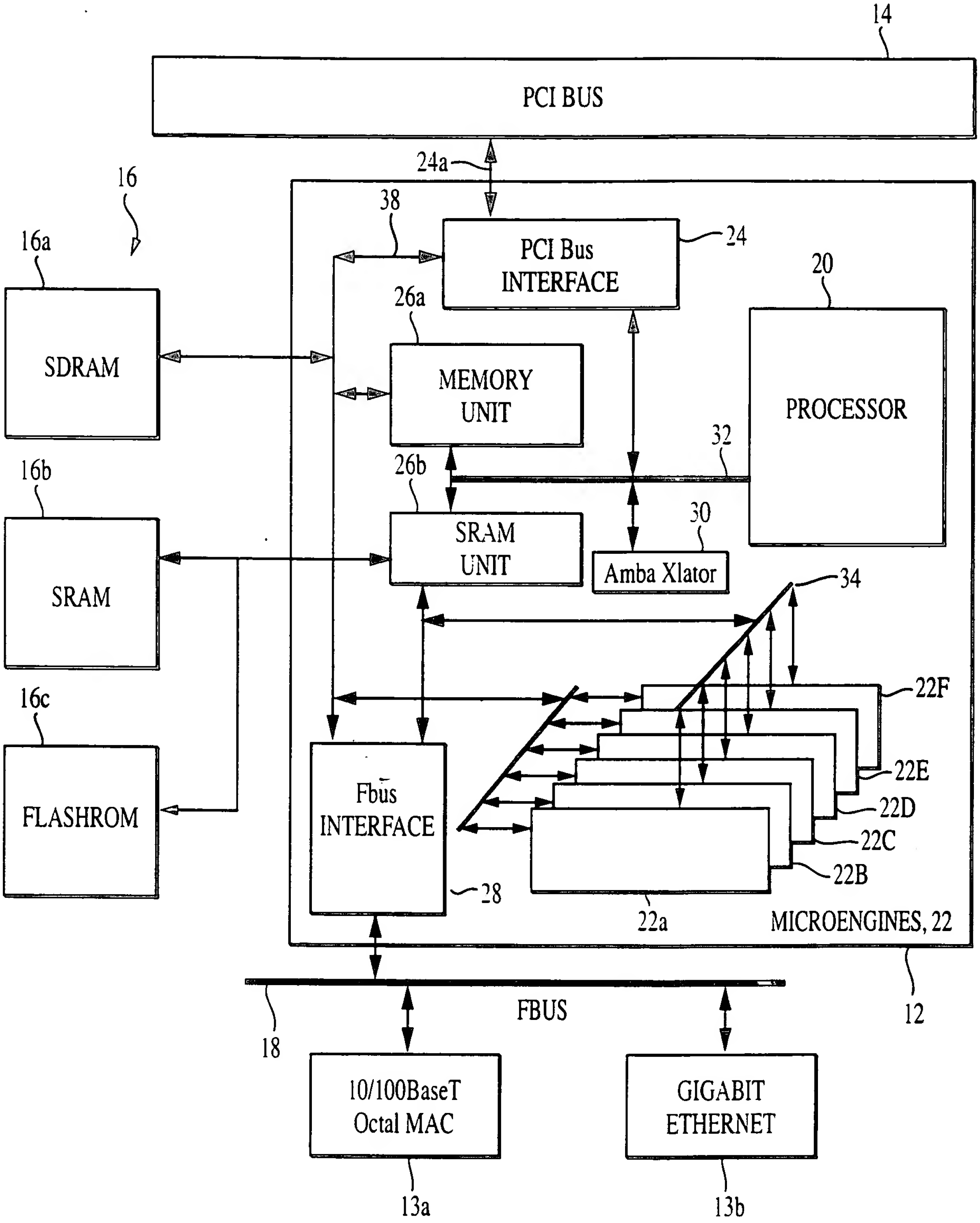
14
PCI BUS
24a
16
38
16a
PCI Bus
INTERFACE
24
20
26a
SDRAM
MEMORY
UNIT
PROCESSOR
32
26b
16b
30
SRAM
UNIT
Amba Xlator
34
SRAM
16c
22F
22E
Fbus
INTERFACE
22D
FLASHROM
22C
22B
28
22a
MICROENGINES, 22
FBUS
12
18
10/100BaseT
Octal MAC
GIGABIT
ETHERNET
13a
13b

FIG. 1

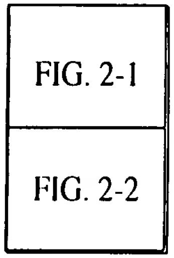
FIG. 2-1
FIG. 2-2

FIG. 2

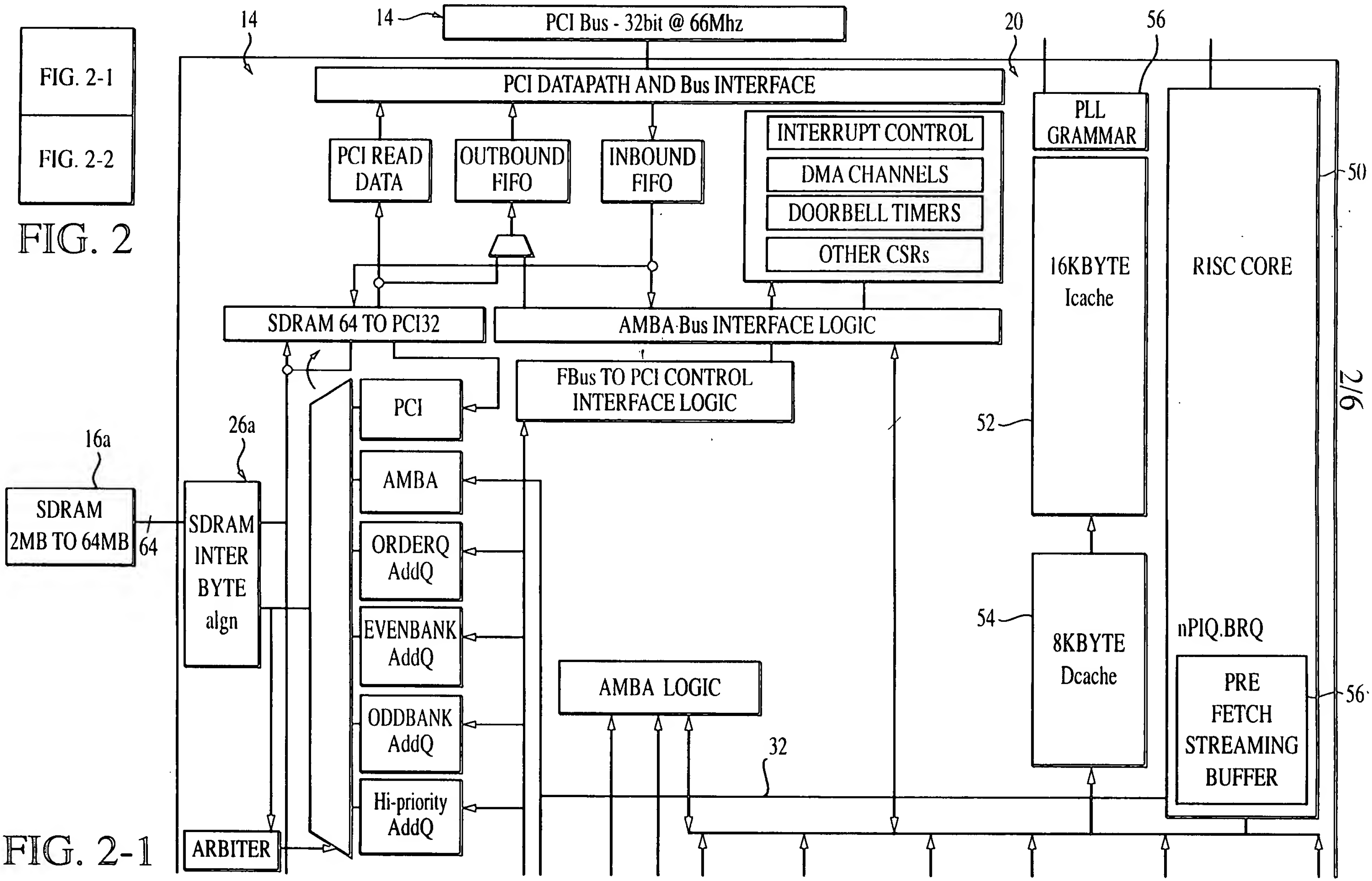
14
14
PCI Bus - 32bit @ 66Mhz
20
56
PCI DATAPATH AND Bus INTERFACE
PCI READ DATA
OUTBOUND FIFO
INBOUND FIFO
INTERRUPT CONTROL
DMA CHANNELS
DOORBELL TIMERS
OTHER CSRs
PLL GRAMMAR
16KBYTE Icache
RISC CORE
50
SDRAM 64 TO PCI32
AMBA-Bus INTERFACE LOGIC
FBus TO PCI CONTROL INTERFACE LOGIC
52
16a
SDRAM 2MB TO 64MB
64
26a
SDRAM INTER BYTE algn
PCI
AMBA
ORDERQ AddQ
EVENBANK AddQ
ODDBANK AddQ
Hi-priority AddQ
ARBITER
AMBA LOGIC
32
54
8KBYTE Dcache
nPIQ.BRQ
PRE FETCH STREAMING BUFFER
56

FIG. 2-1

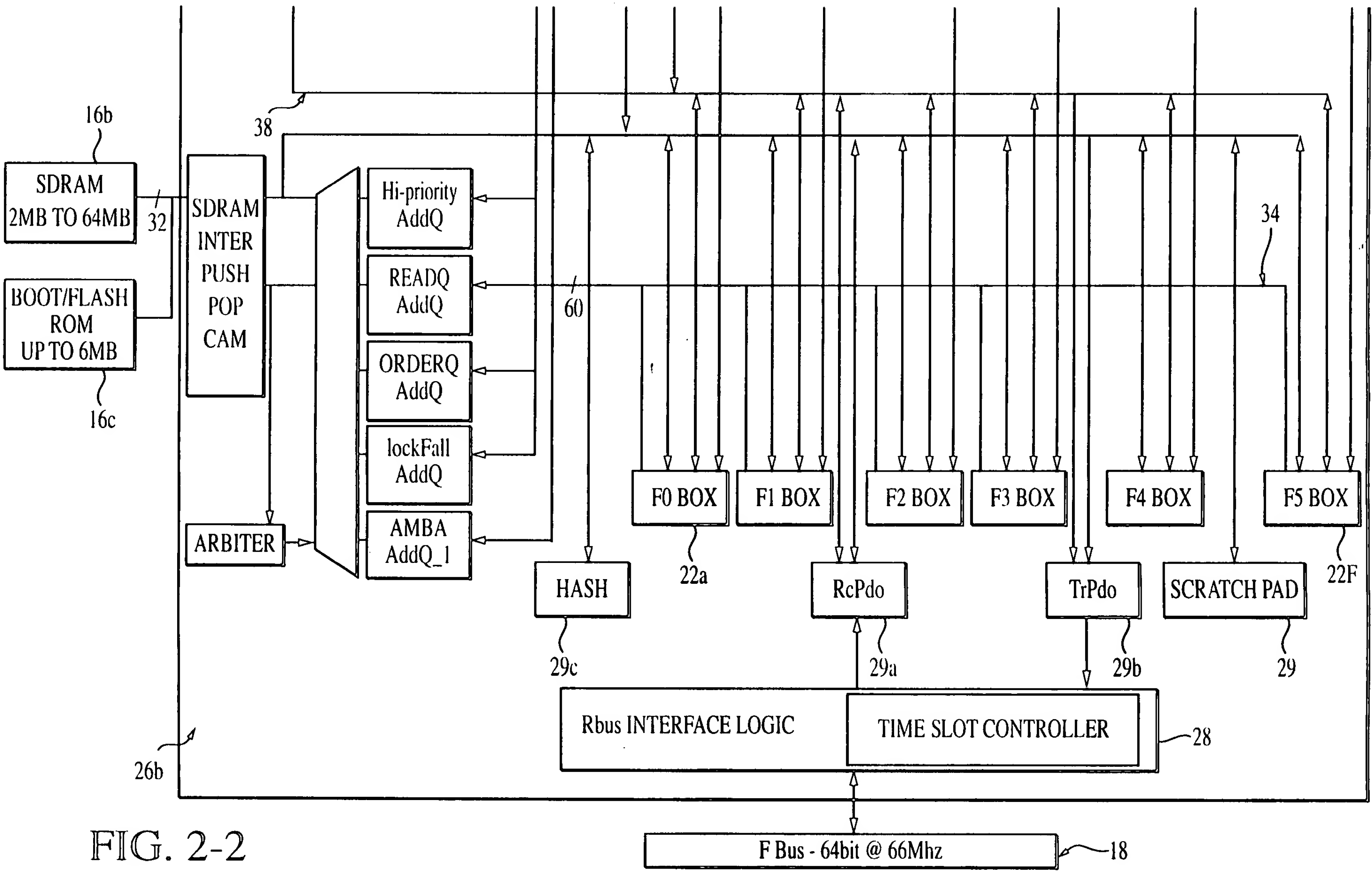
16b
SDRAM
2MB TO 64MB
32
BOOT/FLASH
ROM
UP TO 6MB
16c
38
SDRAM
INTER
PUSH
POP
CAM
ARBITER
Hi-priority
AddQ
READQ
AddQ
ORDERQ
AddQ
lockFall
AddQ
AMBA
AddQ_1
60
34
HASH
29c
F0 BOX
22a
F1 BOX
F2 BOX
F3 BOX
F4 BOX
F5 BOX
22F
RcPdo
29a
TrPdo
29b
SCRATCH PAD
29
Rbus INTERFACE LOGIC
TIME SLOT CONTROLLER
28
26b
F Bus - 64bit @ 66Mhz
18

FIG. 2-2

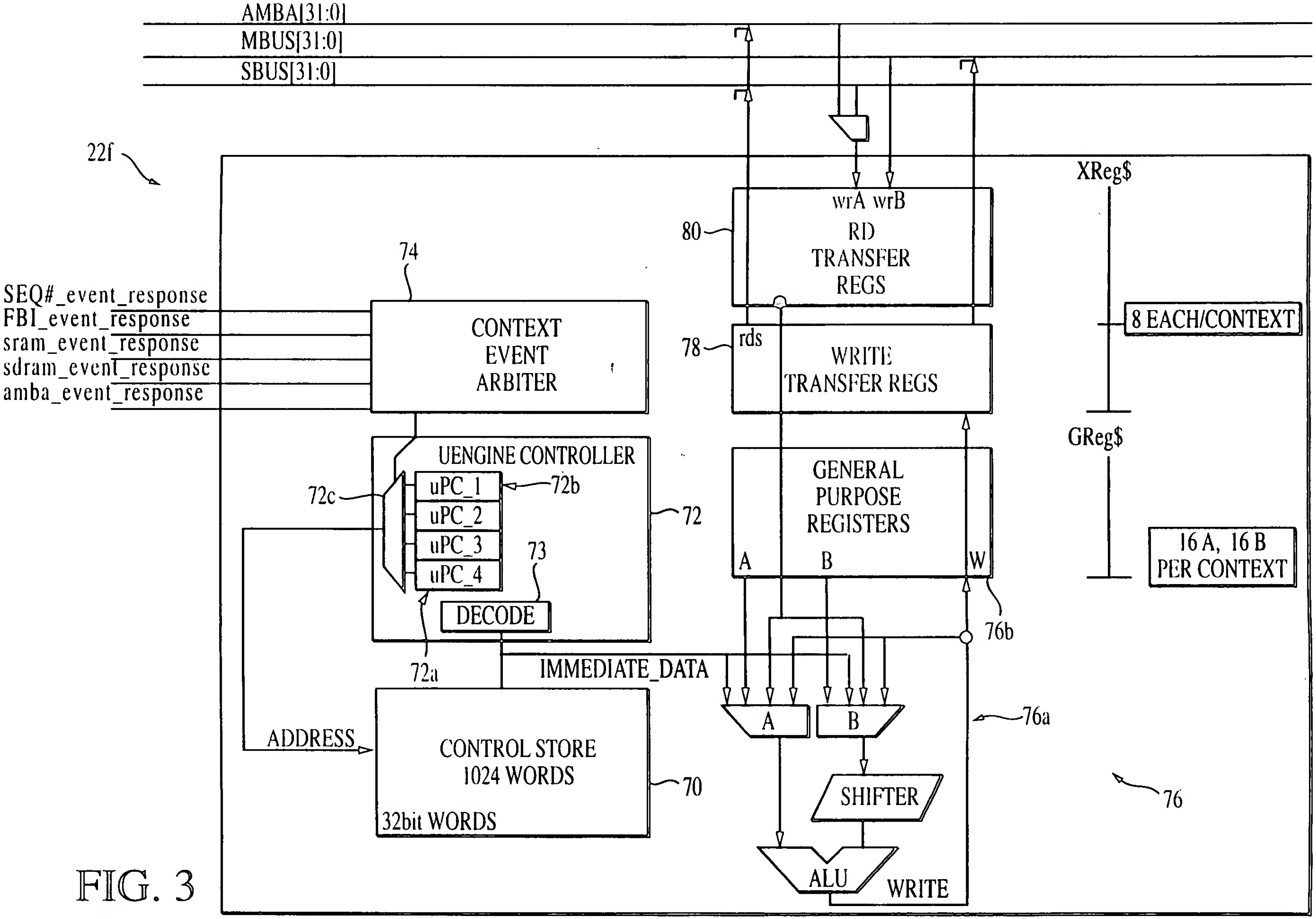
AMBA[31:0]
MBUS[31:0]
SBUS[31:0]
22f
SEQ#_event_response
FBI_event_response
sram_event_response
sdram_event_response
amba_event_response
74
CONTEXT EVENT ARBITER
UENGINE CONTROLLER
72c
uPC_1
uPC_2
uPC_3
uPC_4
72b
72
73
DECODE
72a
IMMEDIATE_DATA
ADDRESS
CONTROL STORE 1024 WORDS
32bit WORDS
70
80
wrA wrB
RD TRANSFER REGS
78
rds
WRITE TRANSFER REGS
GENERAL PURPOSE REGISTERS
A
B
W
76b
A
B
SHIFTER
ALU
WRITE
76a
XReg$
8 EACH/CONTEXT
GReg$
16 A, 16 B PER CONTEXT
76

FIG. 3

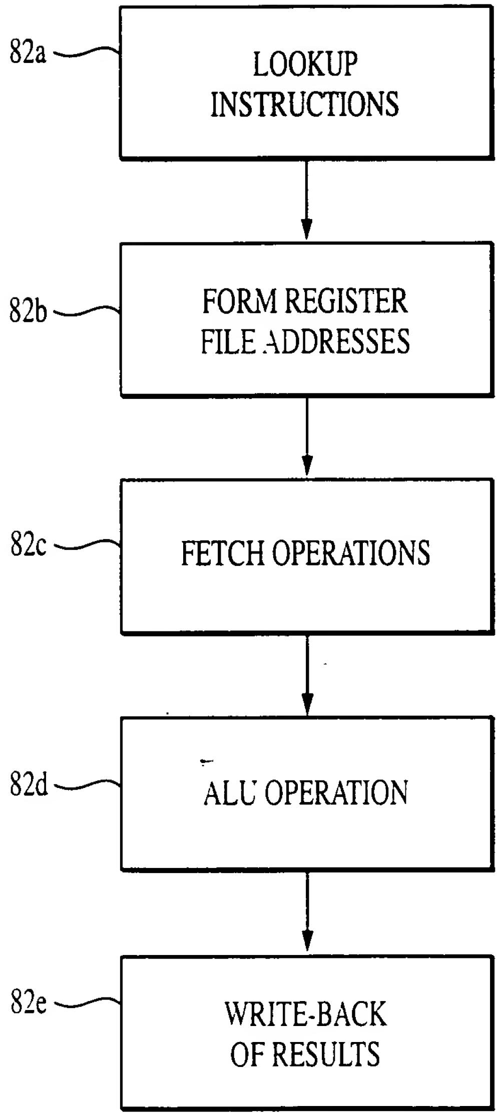
82a
LOOKUP INSTRUCTIONS
82b
FORM REGISTER FILE ADDRESSES
82c
FETCH OPERATIONS
82d
ALU OPERATION
82e
WRITE-BACK OF RESULTS

FIG. 4

<table>
<tr><th></th><th>31</th><th>30</th><th>29</th><th>28 27</th><th>26 25 24 23 22 21</th><th>20</th><th>19 18</th><th>17</th><th>16 15 14 13 12</th><th>11 10 9 8 7</th><th>6</th><th>5</th><th>4</th><th>3 2 1 0</th></tr>
<tr><td>ALU/SHIFT
(set cc)</td><td>0</td><td>0</td><td>sw</td><td>shift</td><td>rel dest reg</td><td colspan="2">amount</td><td>rs</td><td>A rel source</td><td>B rel source</td><td>ro</td><td>im</td><td>Bi</td><td>ALUop</td></tr>
<tr><td>ALU/SHIFT
(set cc)</td><td>0</td><td>0</td><td>sw</td><td>shift</td><td>rel dest reg</td><td colspan="2">amount</td><td></td><td>A rel source</td><td>B rel source</td><td></td><td>I</td><td>O</td><td>ALUop</td></tr>
<tr><td>ALU/SHIFT
(set cc)</td><td>0</td><td>0</td><td>sw</td><td>shift</td><td>rel dest reg</td><td colspan="2">amount</td><td></td><td>A rel source</td><td>immediate</td><td></td><td>I</td><td>I</td><td>ALUop</td></tr>
<tr><td>ALU/SHIFT
(set cc)</td><td>1</td><td>0</td><td>0</td><td colspan="2">dest reg</td><td>sw</td><td colspan="3">A absolute source</td><td>loB Abs Sec</td><td colspan="3">Up B Srl</td><td>ALUop</td></tr>
</table>

Shift Decode:

(rs,r0) decode ([31:0] shifts into [63:32] and take [63;32]):

00 = left rotate
01 = right shift (32-ShfAmt = Right Shift Amt)
10 = left shift
11= double shift (upper A-op shifts into lower B-op)

===> "left rotate" of zero gives zero shift (therwise zero amount signifies indirect shift)

ALU-OP decode:

0000 = B	0100 = ~A&B (~and)	1000 = A-B	1100 = A+B(8)
0001 = ~B	0101 =XOR	1001 = B-A	1101 = A+B(16)
0010 = A&B (and)	0110 = OR	1010 =	1110 = A+B
0011= A&~B (and~)	0111= mul-stuff	1011=	1111= A+B+Cin

FIG. 5